UNIVERSITÉ DE MONTRÉAL

A MULTILEVEL BOTTLENECK ASSIGNMENT APPROACH TO THE BUS DRIVERS'ROSTERING PROBLEM

by

P. Carraresi* and G. Gallo*

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

A MULTILEVEL BOTTLENECK ASSIGNMENT APPROACH TO THE BUS DRIVERS'ROSTERING PROBLEM

by

P. Carraresi* and G. Gallo*

* Istituto di Scienze dell'Informazione
Università di Pisa

Université de Montréal
Centre de recherche sur les transports - Publication #180
Septembre 1980

Dépôt légal - 3e trimestre 1981

ABSTRACT

The problem to find a work assignment for drivers in a given time horizon, in such a way as to have an even distribution of the work load is considered. This problem is formulated as a *Multilevel Bottleneck Assignment Problem* (MBA). Problem MBA is studied: it is shown that it is an NP-Complete problem and an asintotically optimal algorithm is presented. Some computational results are illustrated, which prove the efficiency of the algorithm.

TABLE OF CONTENTS

1. Introduction. 1

2. The multilevel assignment model 2

3. Stable solution for MBA 7

4. An algorithm for MBA. 13

5. The bottleneck assignment algorithm 18

ACKNOWLEDGEMENTS. 21

REFERENCES. 21

1. INTRODUCTION

Most often, when the Bus Drivers Scheduling problem is referred to, only the problem of finding a list of shifts which satisfy the service requirements together with a set of additional constraints is considered. Actually this is only a part of the problem and a second level problem is still to be solved, the rostering problem, that is to assign each shift to one of the drivers.

For some Public Transit Companies the rostering problem is rather easily solved. For instance in most North American Companies, the assignment of the shifts is left to the drivers themselves and the selection is done on a seniority basis.

This actually is not the case in other countries like for instance some of the European countries.

Most often the clauses of the union conventions require that the shifts be assigned to the drivers in such a way that each driver receives an even balance of each type of the shifts during a given time horizon.

In the following a simplified although quite general formulation of this problem is considered. To each shift a weight is assigned, which represents a measure of the cost of the shift for the drivers; such a weight can be simply equal to the length in time of the shift or may take into account other elements like the lateness of the shift, the total elapsed time in the case of split shifts, the time for the driver to arrive at the point on the route where the shift starts, etc.

The problem of finding an even balance of shifts over a given interval of time is then formulated as a bottleneck problem: we want the maximum total weight of the shifts assigned to a single driver to be minimized, that is we want the sum of the weights of the shifts assigned to the driver who is worse off to be minimum.

In particular the problem is formulated as a Multilevel Bottleneck problem. This problem is studied, and an algorithm to find a suboptimal solution is proposed. It is shown that the solutions provided by the proposed algorithm are asintotically optimal. Moreover the results of a rather wide numerical experimentation are given, which proves the efficiency of the algorithm and the goodness of the solutions obtained; both randomly generated problems and real life problems have been used. In the last section a new algorithm for the Bottleneck Assignment problem is presented, which is used as a subroutine to solve the rostering problem. The results of an experimentation to compare such an algorithm with the algorithm due to Derigs and Zimmermann (1978) are illustrated.

2. THE MULTILEVEL ASSIGNMENT MODEL

Let us consider an m days time horizon; for each day n shifts are given. Should the number of the shifts not be the same for all days, then dummy shifts could be added to have an equal number of shifts. Each shift is assigned a real weight; let w_{kj} denote the weight of the j-th shift of day k. The time duration of the shifts or any other measure of the work load of the shift may be used as weight.

For each i and each k we denote by $S_k(i)$ the set of the shifts which can be assigned in day k+1 to the same worker to whom the shift i has been assigned in day k, and by $P_1(i)$ the set of the shifts which can be assigned in day k to the same worker to whom shift i has been assigned in day k+1.

Let us consider the graph of fig. 2.1, where node (k,j) corresponds to the j-th shift of the k-th day.

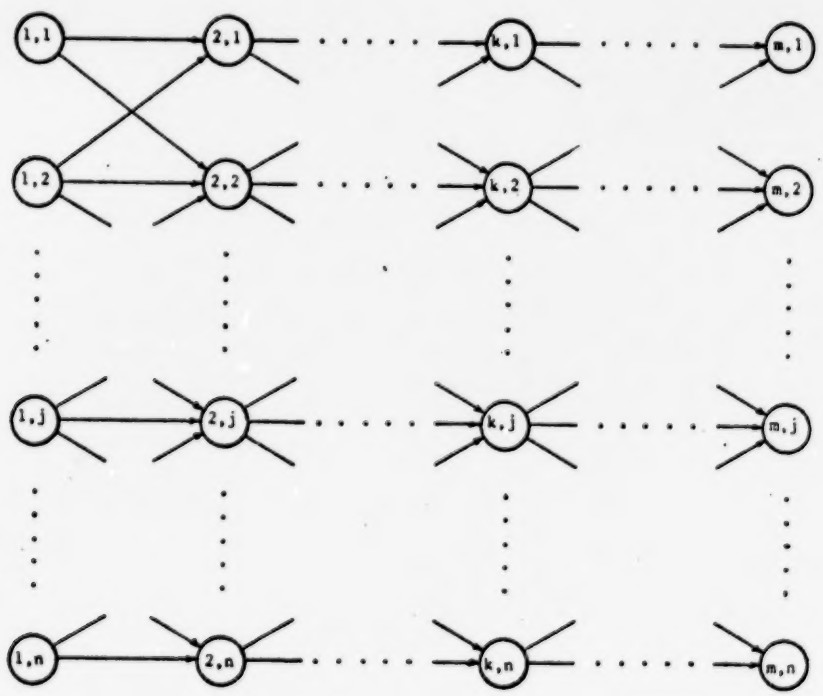

Fig. 2.1

Each node (k,j) has weight w_{kj}. The set $S_k(j)$ corresponds to the set of the successors of node (k,j) in the graph, while $P_k(i)$ corresponds to the set of predecessor nodes. A feasible work assignment to a single

worker corresponds to a path from one of the nodes of the first set, say (1,j) to one of the nodes of the m-th set, say (m,j); its total workload, or length, is given by the sum of the weights of the nodes in the path.

The problem of finding n work assignment such that the maximum workload be minimized, can be formulated as the problem of finding, in the graph of fig. 2.1, n node disjoint paths from the first set of nodes to the last one, such that the longest path has minimum length. If m=2 this is a standard *Bottleneck Assignment* problem. For m>2 we call it a *Multilevel Bottleneck Assignment* problem (MBA). Its mathematical formulation is given next.

$$\text{Min } z$$

$$\text{s.t. a)} \sum_{j \in S_k(i)} x_{ij}^k = 1 \; ; \quad k=1,..,m-i; \quad i=1,..,n;$$

$$\text{b)} \sum_{i \in P_k(j)} x_{ij}^k = 1 \; ; \quad k=1,..,m-1; \quad j=1,..,n;$$

$$\text{c)} \; s_j^1 = w_{1j} \; ; \quad j=1,..,n;$$

$$\text{d)} \; s_j^k = w_{kj} + \sum_{i \in P_{k-1}(j)} s_i^{k-1} x_{ij}^{k-1} \; ; \quad j=1,..,n; \quad k=2,..,m:$$

$$\text{e)} \; s_j^m \leq z \; ; \quad j=1,..,m;$$

$$\text{f)} \; s_{ij}^k \in \{0,1\} \; ; \quad k=1,..,m-1; \quad i,j=1,..,n.$$

PROPOSITION 1. Problem (2.1) has a feasible solution if and only if, for each k=1,..,m-1 the set

$$\sum_k = \{x^k : \sum_{j \in S_k(i)} x_{ij}^k = 1; \quad \sum_{i \in P_k(j)} x_{ij}^k = 1; \quad x_{ij}^k \in \{0,1\}; \quad i,j=1,..,n\}$$

is not empty.

Proof: The proof follows from the fact that constraints from (2.1,c) to (2.1,e) are just definition constraints, and that constraints (2.1,a), (2.1,b) and (2.1,f) define a feasible set Σ which can be partitioned into the m-1 disjoint sets. $\Sigma_1, \Sigma_2, .., \Sigma_{m-1}$.

From Proposition 1, it follows that a feasible solution to (2.1) can easily be determined just by finding the feasible solutions to m-1 assignment problems. It can be worth noting that the problem of finding a feasible solution for MBA when m=3 is a particular case of the 3-Matching problem, which is well known to be NP-complete (Garey and Johnson, 1979). The fact that it can be solved in polynomial time is due to its particular structure.

Unfortunately the problem of finding an optimal solution is not as easy as the problem of finding a feasible one. Actually, as shown by the following proposition, no polynomial time algorithm seems to exist for MBA.

PROPOSITION 2. MBA is NP-complete

Proof: To prove the NP-completeness of MBA we will show that the *Partition Problem* , which is NP-complete (Garey and Johnson, 1979), can be transformed into an MBA problem. Given a set A and a function w: $A \rightarrow Z^+$, the partition problem is to find a partition of A, (A',A"), if it exists (A',A") such that

$$\sum_{a \in A'} w(a) = \sum_{a \in A"} w(a) = 1/2 \sum_{a \in A} w(a).$$

Given an instance of the partition problem with $|A| = m$, build the graph of fig. 2.2:

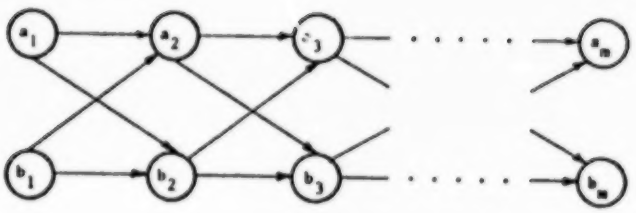

Fig. 2.2

In this graph each node $a_j \epsilon A$ is assigned a weight $w(a_j)$, and each node b_j, j=1,..,m, is assigned a zero weight. If we solve on this graph a Multilevel Bottleneck Assignment we find two paths. Π_1 and Π_2 such that each node a_j,j=1,..,m, belongs to either one of them, and

$$z=\max\{\sum_{a_j\epsilon\Pi_1} w(a_j), \sum_{a_j\epsilon\Pi_2} w(a_j)\} \geq \frac{1}{2} \sum_{a\epsilon A} w(a)$$

Since in MBA one seeks to minimize z, if a solution to the partition problem exists then it is also an optimal solution for MBA. Moreover if in the optimal solution of MBA it is $z > \frac{1}{2} \sum_{a\epsilon A} w(a)$, then no solution for the partition problem exists.//

Remark - Note that if he hold the number of levels m fixed, we cannot state that the problem is still NP-complete. In fact the partition problem on a set of m elements reduces to an MBA problem with m levels. Then it should be possible to find a polynomial time algorithm for MBA when the number of levels is held fixed. This fact is rather interesting since in most pratcical applications the number of levels is 5 or 6, which corresponds to weekly work load assignments.

3. STABLE SOLUTION FOR MBA

Lex (X;z) denote any feasible solution to (2.1), where

$$X = (x^1, x^2, .., x^{m-1}) \quad \text{and} \quad x^k = (x^k_{ij}) \ .$$

Define the vectors s^k and v^k, k=1,.,m:

$$(3.1.a)\begin{cases} s^1_j = w_{1j} \ ; \quad j=1,.,n; \\ s^k_j = w_{kj} + \sum_{i \in P_{k-1}(j)} s^{k-1}_i x^{k-1}_{ij} \ ; \ j=1,.,n; \ k=2,.,m; \end{cases}$$

$$(3.1.b)\begin{cases} v^m_i = 0; \ i=1,.,n; \\ v^k_i = w_{k+1\ i} + \sum_{j \in S_k(i)} v^{k+1}_j x^{k+1}_{ij} \ ; \ i=1,.,n; \ k=m-1,.,1. \end{cases}$$

Clearly $z = \max \{s^m_i : i=1,.,n\} = \max \{v^1_i : i=1,.,n\}$

DEFINITION: A feasible solution (X,z) of (2.1) is a *stable solution* if, for k=1,2,.,m-1, (x^k,z) is an optimal solution of the following Bottleneck Assignment Problem,

$BA_k(X)$:

$$(3.2)\begin{cases} \text{Min } \xi \\ \text{s.t. } \sum_{j \in S_k(i)} \xi_{ij} = 1; \quad i=1,.,n; \\ \qquad \sum_{i \in P_{k-1}(j)} \xi_{ij} = 1; \quad j=1,.,n; \\ \qquad \xi \geq s^k_i + \sum_{j \in S_k(i)} v^k_j \xi_{ij}; \ i=1,.,n; \\ \qquad \xi_{ij} \in \{0,1\}; \ i,j=1,.,n; \end{cases}$$

where s_j^k and v_j^k, j=1,.,n are defined by (3.1.a) and (3.1.b).

It is easy to check that the following proposition holds true:

PROPOSITION 3: Any optimal solution of MBA is a stable solution

Now we give a characterization of the stable solutions for the case in which $S_k(i) = P_k(i) = \{1,.,n\}$ for each i=1,.,n and for each k=1,.,m, that is for the case in which each shift may be followed in a feasible sequence by any shift.

Let Δ be the maximum range of weights for each set of nodes, that is:

$$\Delta = \max_k \{\max_{i,j} (w_{ki} - w_{kj})\} .$$

Then the following theorem provides us with bounds on the error made when, a stable solution is taken in place of the optimal one.

THEOREM 1: Let us consider MBA such that $S_k(i)=P_k(i)$ and $w_{kj} \leq W$ for each i and k; let $(\tilde{X}^*;\tilde{z}^*)$ be an optimal solution and (X;z) be any stable solution, then

$$\tilde{z} - z^* \leq \Delta ,$$

$$\lim_{m \to \infty} \frac{\tilde{z}-z^*}{z^*} = 0.$$

Proof: It is well known (Lawler, 1976) that an optimal solution to (3.1) for the case of complete bipartite graphs, can be obtained as follows:

$$\text{let } s^k_{i_1} \geq s^k_{i_2} \geq,.,\geq s^k_{i_m} \text{ and } v^k_{j_1} \leq v^k_{j_2} \leq,.,\leq v^k_{j_m} \quad m$$

then set $\xi_{i_h j_h}=1$, h=1,.,m and $\xi_{ij}=0$ for all other pairs (i,j)

It is easy to check that any solution obtained in this way satisfies the following conditions

$$(3.3)\quad \xi_{ij}=1 \text{ and } \xi_{rs}=1 \Rightarrow (s_i^k-s_r^k)(v_j^k-v_s^k) \le 0.$$

We can always derive from $(\tilde{X},\tilde{z})$ a new solution $(\hat{X},\tilde{z})$ which satisfies conditions (3.3), where $\hat{X}$ may or may not be different from $\tilde{X}$. Consider this new solution $\hat{X}$ and let

$$s_p^m = \max_i \{s_i^m\} \text{ and } s_q^m = \min_i \{s_i^m\} .$$

We may state that $s_p^m - s_q^m \le \Delta$. Assume that this is not true, that is $s_p^m - s_q^m > \Delta$.

Let $\Pi'=\{(1,p_1),(2,p_2),.,(m-1,p_{m-1}),(m,p)\}$

and $\Pi''=\{(1,q_1),(2,q_2),.,(m-1,q_{m-1})(m,q)\}$

be the paths which, in the solution $(\hat{X},\tilde{z})$, end into nodes (m,p) and (m,q) respectively.

Since the condition (3.3) are satisfied, for each $h\epsilon\{i,.,m-1\}$ either it is

$$a)\quad \sum_{k=1}^{R} (w_{kp_k}-w_{kq_k}) > \Delta \text{ and } \sum_{k=h+1}^{m} (w_{kp_k}-w_{kq_k}) \le 0.$$

or it is

$$b)\quad \sum_{k=1}^{h} (w_{kp_k}-w_{kq_k}) \le 0 \text{ and } \sum_{k=h+1}^{m} (w_{kp_k}-w_{kq_k}) > \Delta.$$

For h=1, because of the definition of Δ, (b) is true. Let r be the first

value of h such that (a) is true, then:

$$\sum_{k=1}^{r-1} (w_{kp_k} - w_{kq_k}) \leq 0$$

$$\sum_{k=1}^{r} (w_{kp_k} - w_{kq_k}) > \Delta$$

and by substraction we get $w_{rp_r} - w_{rq_r} > \Delta$ which contradicts the definition of Δ.

It is:

$$z^* \geq \frac{1}{n} \sum_{j=1}^{n} \sum_{k=1}^{m} w_{kj} = \frac{1}{n} \sum_{j=1}^{n} s_j^m \geq s_q^m$$

and $\tilde{z} = s_q^m$. Then

$$\tilde{z} - z^* \leq s_p^m - s_q^m \leq \Delta.$$

Since $z \leq mW$, one has

$$\lim_{m \to +\infty} \frac{\tilde{z}-z^*}{z^*} \leq \lim_{m \to +\infty} \frac{\Delta}{mW} = 0.$$

the proof is so completed. //

From Theorem 1 it follows that any heuristic algorithm for MBA which finds stable solutions is asintotically optimal; moreover the error is bounded.

Unfortunately Theorem 1 is not valid if for some k the bipartite graph of the single level Bottleneck Assignment (3.2) is not complete. Actually in practice also for these cases the error becomes smaller and smaller as the number of levels increases.

In fig.3.1 the results of some computational experience are reported. Two different sets of 13 problems each have been generated; for both the sets it is: n=50, and m going from 5 to 40. In the first set each bipartite graph is complete (arc density=1), while in the second, the average number of arcs starting from each node is 25 (arc density=.5).

For each problem a stable solution has been determined and the value

$$\varepsilon = \frac{\tilde{z} - \frac{1}{n}\sum_{k=1}^{m}\sum_{j=1}^{n} w_{kj}}{\frac{1}{n}\sum_{k=1}^{n}\sum_{j=1}^{m} w_{kj}} \geq \frac{\tilde{z}-z^*}{z^*}$$

has been computed. The curves of fig. 3.1 suggest that the stable solution are good suboptimal solutions for MBA. This consideration is widely supported by the computational experience which is illustrated in the next section.

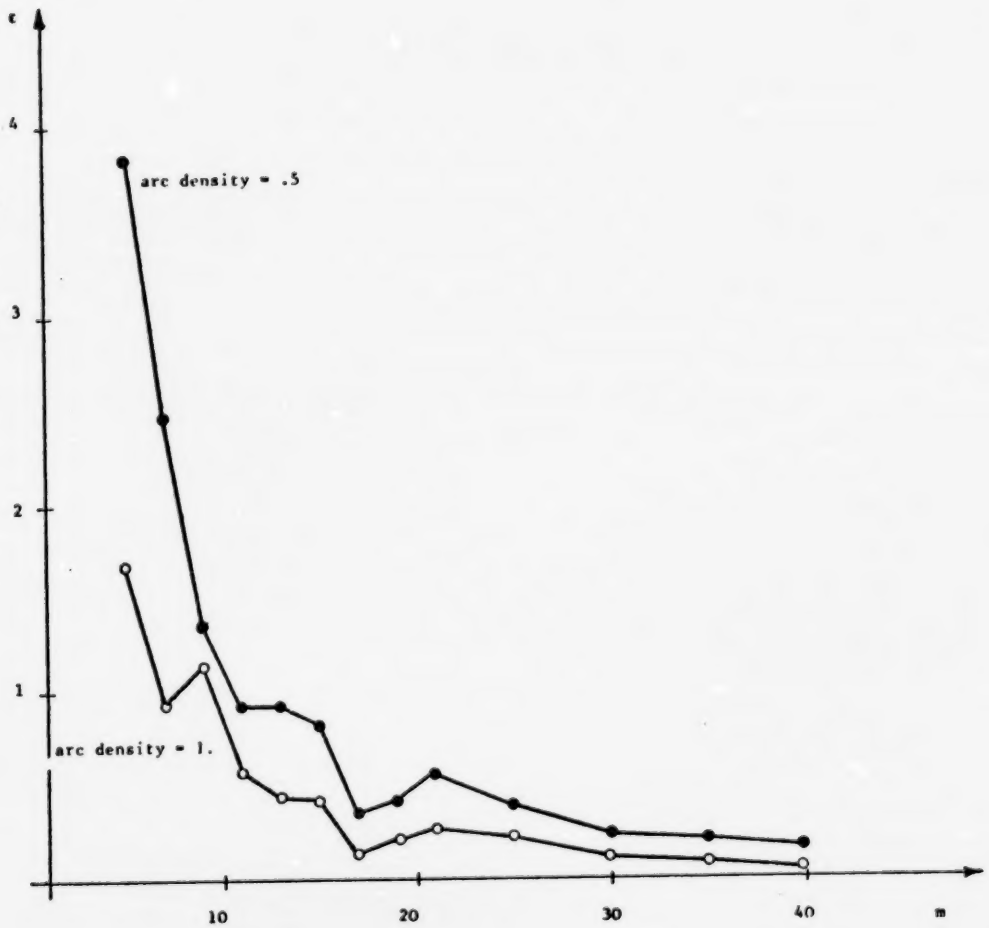

Fig. 3.1

4. AN ALGORITHM FOR MBA

Now we present an algorithm which finds a stable solution for MBA. This algorithm, which we call SSMBA (Stable Solutions for Multilevel Bottleneck Assignment); has two phases. In the first phase a feasible solution is determined; in the second phase such solution is improved until a stable solution is found.

In the first phase we start from the solution $(\bar{X},\bar{z})$ where $\bar{X} = (w^1,w^2,..,w^{m-1})$ with w^k a null vector for each k, and $\bar{z} = 0$.

First problem $BA_i(\bar{X})$ is solved and a feasible assignment between the first two sets of nodes is determined.

Let $\bar{x}^1$ be the vector defining such an assignment and z^1 the value of the objective function; we set $\bar{X} = (\bar{x}^1,w^2,..,w^m)$, $\bar{z} = z^1$, and solve problem $BA_2(\bar{X})$, and so on until, after the solution of m-1 single level bottleneck assignment problems, a feasible solution $(\bar{X};\bar{z})$ is obtained.

Let us call FBA(k,X) a function which given k and X returns (x^k,z) optimal solution of $BA_k(X)$. If problem $BA_k(X)$ has no feasible solution then the function FBA(k,X) returns the solution $(0,+\infty)$.

If for some k no feasible solution exists then no feasible solution exists for MBA and the algorithm stops.

In the second phase the solution $(\bar{X},\bar{z})$ is improbed by solving a sequence of problems of the type $BA_k(\bar{X})$ until a stable solution is found.

A formal description of the algorithm is given next.

Procedure SSMBA

```
Begin   *{First Phase}*
(X;z) := (w^1,w^2,..,w^(m-1);0) 1
for k:=1 to m-1 do
  begin
  (x̄^k,z̄^k) := FBA(k,X) ;
  if z̄^k = + ∞ stop else (X̄;z̄) := (x̄^1,..,x̄^k,w^(k+1),..,w^(m-1);z̄^k)
  end;
repeat  *{Second Phase}*
  flag := 0
  for k:=1  to m-1 do
    begin
      (x̄^k,z̃) := FBA(k,X̄) ;
      if z̃ < z̄ then
        begin
          (X̄;z̄):=(x̄^1,..,x̄^(k-1),x̃^k,x̄^(k+1),..,x̄^(m-1);z̃);
          flag := 1
        end;
  end
until flag = 0 ;
end.
```

The main operation performed by procedure SSMBA is the solution of a single level Bottleneck Assignment; then the efficiency of the algorithm mostly depends on the efficiency of the bottleneck assignment algorithm which is used. One of the most efficient algorithms for this problem is

the algorithm due to Derigs and Zimmerman (1978). This algorithm, like most of the algorithms presented in the literature (Lawler, 1976), is of the dual type; it starts with a solution which is not feasible but satisfies some optimality constraints, which are kept satisfied at each iteration, while the infeasibility is reduced. The algorithm ends when the first feasible solution is found. In our problem, except for the initialization phase, a feasible solution is always known; then a primal type algorithm seems to be more appropriate. Moreover the particular structure of the BA problems solved suggests that *ad hoc* algorithms might perform better than general ones. In view of these considerations, a new algorithm, PBM (Primal Bottleneck Assignment), has been designed. In the following section this algorithm is described and an experimental comparison with the algorithm due to Derigs and Zimmermann is presented.

Algorithm SSMBA has been tested both on randomly generated problems and on real life problems.

Six sets of problems have been generated with n=100, m=5. Each set of problems is characterized by a given value of arc density: 0.1, 0.3, 0.5, 0.7, 0.9 and 1.,

In table 1 the results of the experiments on these problems are resumed; each figure is the average over five runs.

By z_1 we denote the objective value of the solution produced by phase 1, by z_2 we denote the objective value of the final solution, while by z* we denote the optimal value of the objective function. Actually the problems have been generated in such a way that a feasible solution always exists,

and that the optimal solution is known. This has been accomplished by properly generating the arcs, and by giving to the nodes in the first four sets weights uniformly distributed between 1 and 100 and to the nodes in the last set weights chosen in such a way that one solution always exists with all the paths having equal length; clearly such solution is an optimal solution.

Finally T is the average CPU time, in sec. x 10^{-2}, to solve MBA and T_2 is the average time of phase 2, while t_1 and t_2 are the average time to solve a single BA in the first phase and in the second phase respectively.

Arc density	$\frac{z_2-z^*}{z^*}$ 100	$\frac{z_1-z_2}{z^*}$ 100	T	T_2	Average # of BA solved	t_1	t_2
.1	11.04	15.38	203.8	153.	21.6	12.69	8.65
.3	8.33	21.68	340.8	268.4	27.4	16.1	11.68
.5	3.22	25.53	452.6	358.4	30.	23.55	13.99
.7	3.88	25.65	437.4	337.4	21.8	25.	19.37
.9	3.56	29.25	621.2	550.6	28.6	17.65	23.51
1.	4.8	27.68	570.6	511.	24.	14.9	26.58

TABLE 1

The results of table 1, in accordance with the ones illustrated in fig.3.1, show that, also for intermediate values of arc density, the proposed algorithm yields good suboptimal solutions.

It can be noted that the results for the case in which the arc density is 1. are much worse than expected. In facts in this case the bipartite graph for each BA problem is complete and its solutions should be particularly easy. The high solution times for the BA problems shown by

the table in this case are due to the fact that the algorithm instead of solving the problems from scratch makes use of the previous found solution as a starting solution. This feature, which is very effective when the density is low, slows down the algorithm in the case of higher valuers of arc density. Should the algorithm only be used with problems with high arc density, it should be slightly modified in such a way as to have the BA routine starting from scratch also in phase 2.

In the practical problem on which SSMBA has been used the arc density always had values far below 1.

Algorithm SSMBA has also been tested on a real life problem. The problem was to determine, starting from a given set of shifts, the weekly assignments for the drivers. Each weekly assignment consisted in a sequence of five shifts. Seven distinct types of sequences have been considered depending on the starting and on the ending days: weekdays (W), Saturday (S) and Sunday (H). Of each type 50 sequences have been generated.

The results are illustrated in table 2. In first column the type of weekly assignment is indicated. Columns 2 to 4 give information on the length of the shift used to build up the assignment: minimum length, average length and maximum length, all in minutes. The 5-th and the 6-th columns give the minimum length and the maximum length respectively for each type of sequence.

These results confirm the effectiveness of SSMBA: actually starting from shifts with a difference in length of about one hour, sequences with differences from 1 to 4 minutes have been obtained.

Type of sequence	Shifts length (minutes)			Weekly assignment length (minutes)	
	Minimum	Average	Maximum	Minimum	Maximum
HWWWW	334	376	400	1876	1878
SHWWW	334	375	400	1872	1878
WSHWW	333	372	400	1859	1861
WWWSH	328	373	399	1865	1867
WWWWS	328	375	399	1872	1876
WWWWW	328	374	400	1869	1870

TABLE 2

5. THE BOTTLENECK ASSIGNMENT ALGORITHM

In the present section a broad description of algorithm PBA is given.

In the following we denote by N_1 and N_2 the two sets of nodes of the bipartite graph, by a_i $i \in N_1$, and by b_j, $j \in N_2$, the weights of the nodes. We assume that the nodes are ordered in such a way that $a_1 \geq a_2 \geq \ldots \geq a_n$ and $b_1 \leq b_2 \leq \ldots \leq b_n$, where $n=|N_1|$, and that a feasible solution y is known. The algorithm performs the following main operations:

1. Find a bottleneck arc, that is an arc (r,s) such that

   $a_r+b_s = \max\{a_i+b_j : y_{ij}=1\}$.

2. Find a decreasing alternating cycle starting from node r, that is a simple cycle $(i_1,j_1;i_2,j_2,..,i_p,j_p)$ such that:

   i) $i_1 = r, \quad j_p = s;$

   ii) $a_{i_h} + b_{j_h} \leq a_r + b_s, \quad h=1,..,p;$

   iii) $\bar{y}_{i_h j_h} = 1, \quad h=2,..,p.$

   (To perform this step a labeling technique is used, which takes advantage of the fact that the nodes are sorted). If no decreasing

alternating cycle exists, stop; $\hat{y}$ is an optimal solution.

3. Set $\bar{y}_{rs}=0$, $\bar{y}_{i_h j_{h-1}}=0$ for $h=2,..,p$ and $y_{i_h j_h}=1$ for $h=1,...,p$.
   Return to step 1.

At the beginning a sorting routine is used to give the proper ordering to the nodes. An initialization phase is needed in order to use the algorithm when no feasible solution is given, like when it is used in the first phase of SSMBA.

In the initialization phase the set N_2 is enlarged by adding the dummy nodes n+1, n+2,..,2n with weights $b_{n+1}=b_{n+2}\ldots=b_{2n}=+\infty$. The vector is enlarged too in order to contain the components $y_{1n+1}, y_{2n+2},..,y_{n2n}$. Then for each i we set $\bar{y}_{ij(i)}=1$, where

$$j(i)=\min\{j:j\in S(i), \sum_{\ell<i}\bar{y}_{\ell j}=0\}, \text{ and } \bar{y}_{ij}=0, \ \forall\ j\neq j(i).$$

Note that S(i), the set of the successors of i contains at least the node n+i. Clearly if the bipartite graph is complete the vector $\bar{y}$ is an optimal solution.

If an index i exists such that $\bar{y}_{in+1}=1$, then $\bar{y}$ is not feasible for BA. The feasibility can be achieved by making use of the same procedure described before with the following minor modification: in step 2 an alternating path is determined instead of the cycle. The new step 2 is:

2. Find an alternating path starting from node r, that is a simple path $(i_1,j_1,i_2,j_2,..,i_p,j_p)$ such that

   i) $i_1=r$;

   ii) p is such that $\sum_i \bar{y}_{i_p j_p}=0$;

iii) $y_{i_h j_{h-1}} = 1$, $h=2,..,p$.

If no alternating path exists, stop: no feasible solution exists.

Note that, unless y is feasible, the bottleneck arc is of the type (r,n+r). The first phase ends when $\max \{a_i + b_j\, y_{ij} + 1\} < +\infty$

Algorithm PBA has been compared with the algorithm of Derigs and Zimmerman, (DZ), on a set of randomly generated problems with n=100. The problems have been generated with different values of arc density and the costs with uniform distribution within given ranges.

For each arc density and each cost range, ten random problems have been generated and solved by means of both algorithms. The average CPU time over the ten runs is indicated in table 3: the top figure gives the time with PBA, the lower the time with DZ. Times are in seconds x 10^{-2}; an IBM 370/168 computer has been used.

The results show that PBA is much faster than DZ. It can be interesting to note that the problems with an intermediate value of density seem to be the most difficult ones for PBA, while the running time of DZ increases for increasing values of arc density.

REMARK: In order to have an unbiased comparison the Fortran implementation of DZ which is given in Derigs and Zimmerman (1978) has been used.

Arc density	Weight range 1 - 10	1 - 100	1 - 1000
.1	8.5 17.4	9.9 17.3	11.5 27.3
.3	11.1 14.6	15.6 26.8	16.2 40.4
.5	12.8 26.6	18.4 58.7	19.8 63.8
.7	10.5 17.	11.8 58.7	16.4 79.8
.9	7.9 30.5	9.5 73.2	16.8 96.5

TABLE 3

ACKNOWLEDGEMENTS

The authors are grateful to P. Camerini for having suggested the proof of proposition 2.

REFERENCES

LAWLER, E.L. "Combinatorial Optimization: Networks and Matroids", Holt, Rinehart and Winston (1976).

GAREY, M.R. and JOHNSON, D.S. "Computers and Intractability. A Guide to the Theory of NP-Completeness", W.H. Freeman and Company (1979).

DERIGS, U. and ZIMMERMANN, U. "An Augmenting Path Method for Solving Linear Bottleneck Assignment Problems", Computing 19, 285-295 (1978).